International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 12, Issue 12 (December 2016), PP.20-26

An Approach for Assessing Reusability of Component-Based Software (CBS)

Reena[1], Pradeep Kumar Bhatia[1]
[1]*Department Of Computer Science Engineering Guru Jambheshwar University Of Science & Technology, Hisar (Haryana)*

Abstract: Component Based Software Engineering (CBSE) is a paradigm that emphasizes on the design and construction of computer- based system using reusable software components because reusability is an effective way to improve productivity in CBSD. This paper proposes a modified reusability metrics suite and a new technique for analyzing the reusability of component adopted for component based system. Also, it introduces new COTS metrics for CBSE system using CK metrics.
Keywords: Reusability, CBSE, Quality, OO metrics.

I. INTRODUCTION

In the present scenario, software enhances our daily life so there is probably no other human – made material which is omnipresent than software in the society. It has become an important part of many aspects of society: home appliances, telecommunications, automobiles, airplanes, and web teaching so on. Science and technology demand high quality software for making improvements and breakthroughs so software reuse is currently one of the most active and creative research areas in computer science [5]. Software reuse provides the key to enormous savings and benefits in software development; the U.S. department of defense alone could save $300 million annually by increasing its level of reuse by as little as 1%. We have yet to identify a reliable way to quantify what we mean by "reusable" software. Such a measure may help us only to learn how to build reusable components but also to identify reusable component among the wealth of existing programs. These existing programs contain years of knowledge and experience gained from working in the application domain and meeting the organization's software needs. If we could extract this information efficiently, we could gain a valuable resource upon which to build future applications for society. Unfortunately, working with existing software poses several significant problems due to its inconsistent quality, style, documentation and design so we need an economical way to find domain knowledge and useful artifacts within these programs [4]. However, in spite of all such advantage of reuse, there are so many time, reuse may lead to the software failure. The main reason is due to the lack of experience for reuse, lack of documentation, tool and methodology for reuse along with several others. Software reusability is an effective way to enhance productivity. Software reusability development differs from the traditional way of software development in that it affects software measure, therefore new way of software reusability metric is needed, and however new model of software reusability is needed to be established. In order to improve the quality and reusability of component, reusability model for software component is studied in order to given scientific and accurate evaluation standard but the component metrics alone are not sufficient for an integrated environment, because there is a need to measure the stability and adaptability of each component when it is integrated with other components.

Software development cost estimation has been the subject of vigorous study over the past few decades. Software reusability provides opportunity for reducing the overall development cost and time of the software and improving the overall quality. Although, significant progress has been made in the areas of reusability. The growing trend towards using CBSE approach to include COTS component has taken the original concept of reuse into a completely different arena. The use of Commercial-Off-The-Shelf, software components is increasing in today's software development environment. Component-based reuse is widely accepted as an important reuse strategy and component-based reuse programs heavily depend on software reuse repositories for achieving success [1,2,3].

Component-based software engineering (CBSE) is an approach to software development that relies on software reuse. It emerges from the failure of object-oriented development to support effective reuse. Components are more abstract than object classes and can be considered to be stand-alone service providers. Apart from the benefits of reuse, CBSE is based on sound software engineering design principles [1,2]:

- Components are independent so do not interfere with each other;
- Component implementations are hidden;
- Communication is through well-defined interfaces;
- Component platforms are shared and reduce development costs.

CBSE is a reuse-based approach to defining and implementing loosely coupled components into systems.

1.1 Characteristics of a component

- A component is a software unit whose functionality and dependencies are completely defined by its interfaces
- A component model defines a set of standards that component providers and composers should follow
- During the CBSE process, the processes of requirements engineering and System design are interleaved
- Component composition is the process of 'wiring' components together to create a system.
- When composing reusable components, you normally have to write adaptors to reconcile different component interfaces
- When choosing compositions, you have to consider required functionality, non-functional requirements and system evolution

Component-based software engineering is successful if the reusable and certified component available for development of CBSS and decreases the overall cost as well as improve the quality as compared to traditional software development. Cost and quality plays a very important role at every phase in developing software and if cost can be decreased, a lot can be gained. The broader context of discussion is the cost associated with the different stages of software development. It is found that reusability play an important role in estimating cost savings and quality improvement [19].

According to Danial et. al. [4], the core measurement factors are given below that can influence the reusability of component:

- **Availability:** The availability of a software component can determine how easy and fast is to retrieve it and how much percentage it fulfil the given requirement.
- **Documentation:** A good documentation can make the software component more reliable since it makes it easier to understand and use.
- **Complexity:** The complexity of a software component determines how usable it is and how easy it is to adapt the software component in the new context of use.
- **Quality:** The quality of the component or module directly determines how usable it is in a given context. The quality of a component is regarded as a characteristic which describes how good it fulfils its requirements and also how error- and bug-free it is.
- **Maintainability:** Maintainability of a software component can determines how usable it is for reuse. After the integration into the new system, the component should be able to adjust to the changes in the system along with new requirement.
- **Adaptability:** The adaptability is the ease with which a component can be adapted to fulfil a requirement that differs from that for which it was originally developed [4].
- **Reuse:** Reuse of the component can also be used to infer how usable and how easy it is to adapt it. The amount and frequency of reuse, especially in contexts similar to that of the developer can serve as reference point.
- **Price:** Price of the software component determines how expensive it is to reuse.

Software reuse is only relevant when it has positive economical impacts in organizations and cost estimation models are fundamental in assessing these impacts. Component-based reuse is widely accepted as an important reuse strategy and component-based reuse programs heavily depend on software reuse repositories for achieving success [1,3,5], however the main focus on the reuse repository area is on classification and retrieval problems. Existing repositories have their efficiency measured by a sum of features [7], but lack mechanisms to automate the task of collecting and maintaining trace links between reused components and their clients. There are various ways to achieve reusability. Many researchers have specified the benefits of reuse in their reports [15,16]. Developing software from scratch is the most complex part of the process. Reusability reduces this need and hence allows a greater focus on quality. Investments in information technology by U.S. businesses continue to spiral, with estimates topping $400 billion for 1997 [8]. The software development process in many organizations has been associated with cost and quality. There should be some Economic models quantify the costs and benefits of reusability. Several authors have modified the cost models that are today used to estimate time and effort and for the development both of components and of applications using components [9] [10][11] In this paper we investigate reusable metric for CBS to measure the extent of reusability aspect that calculate the cost to make component reusable and savings in avoided work which will eventually reduced the overall cost of software development

This paper is organized as follows. In the next section, we discuss various existing OO metrics developed by various researchers. In section 3 we propose various **Component** Based System (CBS) Metrics**.** In section 4 we propose Reusability assessment of component based system (CBS). In section 5, we draw some conclusions and give directions for future work.

II. METRICS

CK metrics is a suite of metrics for OO design, and is composed of six design metrics [17,18]:
2.1 Weighted Methods per Class (WMC)
2.2 Depth of Inheritance Tree (DIT)
2.3 Number of Children (NOC)
2.4 Coupling between Object Classes (CBO)
2.5 Response for a Class (RFC)
2.6 Lack of Cohesion in Methods (LCOM)

2.1 Weighted Method per Class (WMC)

WMC = number of methods defined in class
WMC is a count of the methods implemented within a class or the sum of complexities of the methods.

2.2 Depth of Inheritance Tree (DIT)

DIT = maximum inheritance path from the class node to the root of the tree
The deeper a class within hierarchy, greater the number of methods and variables it is likely to inherit, making it more complex.

2.3 Number of Children (NOC)

NOC = the number of immediate sub-classes of a class
Number of children is equals to the number of immediate child classes derived from a base class.

2.4 Coupling between Object Classes (CBO)

CBO = the number of classes to which a class is coupled
Two classes are coupled when methods declared in one class use methods or the instance variables defined by the other class.

2.5 Response for Class (RFC)

RFC= M + R (First-step measure)
RFC' = M + R' (Full measure)
M =the number of methods in the class
R = the number of remote methods directly called by methods of the class
R' = the number of remote methods called, recursively through the entire call tree

A message is a request that an object makes to another object to perform an operation and it is executed as a result of receiving a message is called a method.

RFC is the total number of all methods within a set that can be called in response to message sent to an object. This includes all methods accessible within the class. RFC is used to check the class complexity. If the number of methods larger than the complexity of the class is also increase [12].

2.6 Lack of Cohesion of Methods (LCOM)

LCOM uses variable to measure degree of similarity between methods. In this, we can measure the cohesion for each data field in a class; calculate the percentage of method that uses the data field.

III. COMPONENT METRICS

3.1 Expansion for NOM: weighted for components and number of classes

A component made up of a group of classes, so it is rational that the complexities of the various classes affect the complexity of the emerging component. If the classes are complex, then they are more difficult to understand and maintain consequently the component will be more complex and difficult to maintain [9]. Utilizing NOM to measure the complexity of the classes, here we define the weighted class for a component (WCC) as:

$$\mathrm{WCC} = \sum_{i=1}^{m} \mathrm{NOM}(C_i)$$

For n components

$$\sum_{i=1}^{n} \mathrm{WCC} = \sum_{j=1}^{n} \sum_{i=1}^{m} \mathrm{NOM}(C_{ji})$$

3.2 Expansion for DIT: maximum of the DIT and mean of unrelated trees

If the values of DIT classes higher, then it is hard to maintain, so effort in maintaining group of classes can therefore be indicated by the value DIT. In particular we beginning consider both the highest value of DIT, MAXDIT.

$$\text{MAXDIT} = [\max\{\text{DIT}(C_i)\}, i = 1 \text{ to } m]$$

For n components

$$\text{MAXDIT} = [\max\{\max\{\text{DIT}(C_i)\}, i = 1 \text{ to } m]_{j=1,n}$$

3.3 Expansion for NOC: number of children for component

NOC is expended by counting the number of children of all the classes in the component, that's why we call this metric, number of children for a component (NOCC).

$$\text{NOCC} = \sum_{i=1}^{m} \text{NOC}(C_i)$$

For n components

$$\sum_{i=1}^{n} \text{NOCC} = \sum_{j=1}^{n} \sum_{i=1}^{m} \text{NOC}(C_{ji})$$

3.4 Expansion for CBO: external CBO

Here we define level of coupling for more than one component. External CBO is the number of external classes coupled to it.

$$\text{EXTCBO} = \sum_{i=1}^{m} e_i$$

For n components

$$\sum_{i=1}^{n} \text{EXTCBO} = \sum_{j=1}^{n} \sum_{i=1}^{m} e_{ji}$$

Where e_i is the number of external classes integrated to the class Ci.

3.5 Expansion for RFC: response set for the component

The response set of components (RFCOM) is the number of all the methods in the member classes and the methods invoked by those classes. This value is the sum of the values of RFC for all the classes in the set of component.

$$\text{RFCOM} = \text{LCOM} = \sum_{i=1}^{m} \text{RFC}(C_i)$$

$$\sum_{i=1}^{n} \text{RFCOM} = \text{LCOM} = \sum_{j=1}^{n} \sum_{i=1}^{m} \text{RFC}(C_{ji})$$

IV. RUSABILIITY ASSESSMENT OF CBS

4.1 Reusability assessment of Object-oriented System

Definition of RFC metric leads to fact that:

RFC = Number of local methods + Number of invoked methods from other classes

= NOL + NOC …… (1)

If all method complexities in a class are considered to be unity,

Then WMC=n, where n is the number methods in the class implies NOL=WMC.

By definition of CBO metric; the objects are accessed internally through remote method and instance variable that is:

CBO= Number of remote methods + number of instance variables

Or
CBO = NOC + number of instance variables

Remote methods are much larger than instance variable, we may reject instance variables, this implies
CBO≅ NOC
Substituting the metrics WMC and CBO for NOL and NOC respectively in equation (1) then
RFC ~ WMC +CBO……..... (2)
On examining equation (2):
If WMC and CBO both increase then RFC also increase, and if WMC and CBO decrease then RFC also decrease.
In case of high reusable code , classes in a system are loosely coupled then CBO metric will be low and increase in RFC is due to WMC that is including more methods in class increase the RFC metric but since coupling is basic, the CBO metric should not significantly increase. Equation RFC= NOL + NOC approaches to RFC=WMC (i. e. NOL + NOC → WMC).
In case of low reusable code, classes in a system are highly coupled then the CBO metric will be high and increase in RFC would be both CBO and WMC. Equation RFC=NOL + NOC approach to RFC= WMC+CBO (i.e. NOL +NOC → WMC+CBO).

Table1. Reusability Assessment for classes in OO system.

Reusability	Coupling Between Objects (CBO)	RFC = NOL + NOC approaches to
HIGH	Decrease	RFC = WMC
LOW	Increase	RFC = WMC + CBO

4.2 Reusability assessment of a Component
On the same outline of equation (2) REFCOM metric of a component leads to fact that:

$$\sum_{i=1}^{n} \mathrm{RFC}(C_i) = \sum_{i=1}^{n} \mathrm{WMC}(C_i) + \sum_{i=1}^{m} e_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

or
RFCOM = WCC+ EXTCBO…… (4)

Equation (4) shall be used to identify reusability of a component. On examining it:
If WCC and EXTCBO both increase then RFCOM also increase, and if WCC and EXTCBO decrease then RFCOM also decrease.

In case of high reusable code , classes in a component are loosely coupled then EXTCBO metric will be low and increase in RFCOM is due to WCC that is including more methods in classes of a component increase the RFCOM metric but since coupling is basic, the EXTCBO metric should not significantly increase. Equation RFCOM= WCC+ EXTCBO approaches to RFCOM=WCC.

In case of low reusable code, classes in a component are highly coupled then the EXTCBO metric will be high and increase in RFCOM would be both EXTCBO and WCC. Equation RFCOM= WCC+ EXTCBO.

Table 2. Reusability Assessment of a component

Reusability	Coupling within component (Intra-component coupling)	N RFCOM =∑ RFC (Ci) i=1
HIGH	Decrease	RFC =WCC
LOW	Increase	RFC = WCC+ EXTCBO

4.3 Reusability assessment of Component Based System
On the same outline of equation (4) CBS_RFC metric of a CBS leads to fact that:

$$\sum_{i=1}^{m} \mathrm{RFCOM} = \sum_{i=1}^{m} \mathrm{WCC} + \sum_{i=1}^{m} \mathrm{EXTCBO} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Or

CBS_RFC = CBS_WMC + CBS_CBO

Equation (5) shall be used to identify reusability of a component based system. On examining this equation: If CBS_WMC and CBS_CBO both increase then CBS_RFC also increase, and if CBS_WMC and CBS_CBO decrease then CBS_RFC also decreases.

In case of high reusable CBS , components in a CBS are loosely coupled then EXTCBO metric will be low and increase in CBS_RFC is due to CBS_WMC that is including more methods in classes within each component increase the CBS_RFC metric but since coupling is basic, the CBS_CBO metric should not significantly increase. Equation CBS_RFC = CBS_WMC + CBS_CBO approaches to CBS_RFC = CBS_WMC In case of low reusable CBS, components in a CBS are highly coupled thus components are highly coupled then the CBS_CBO metric will be high and increase in CBS_RFC would be both CBS_CBO and CBS_WMC .

Table3. Reusability for inter-components

Reusability	Coupling Between Components (Inter -component coupling)	CBS_RFC=CBS_WMC + CBS_CBO
HIGH	Decrease	CBS_RFC = CBS_WMC
LOW	Increase	CBS_RFC=CBS_WMC + CBS_CBO

V. CONCLUSION

Software reuse is only relevant when it has positive economical impacts in organizations. The proposed CBSE metric estimates the effect of component reusability in the overall development process. In this paper, various OO metrics and Component metrics have been illustrated. Also, reusability assessment is done for Object oriented components and inter components. As a future work, an empirical study will be conducted to assess reusability of different components in component based system.

REFERENCES

[1]. S. Henninger (1997), "An Evolutionary Approach to Constructing Effective Software Reuse Repositories.", ACM Transactions on Software Engineering and Methodology, Vol 6, No 2, pp 111-40.

[2]. R. D. Banker, R. J. Kauffman, D. Zweig(1993), "Repository Evaluation of Software Reuse", IEEE Transactions on Software Engineering, Vol. 19, No. 4, pp. 379-389.

[3]. J. Guo, Luqui(2000) "A Survey of Software Reuse Repositories", 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 92-100.

[4]. V. Lakshmi Narasimhan, P. T. Parthasarathy and M. Das, "Evaluation of a suite for Component Based Software Engineering", vol. 6, pp.731-740, 2009.

[5]. Ivica Crnkovic, "Component-based Software Engineering- New Challenge in Software Development", Proceedings of the 25th International conference on Information Technology interface, pp. 9-18, September 2003.

[6]. Gurdev Singh, Dilbag Singh and Vikram Singh, " A study of Software Metrics" International Journal of Computational Engineering and Management, vol. 11, pp. 476-492, January 2011.

[7]. Pradeep Bhatia, Yogesh Singh and H. L. Verma, "An Empirical Study for Assessing Quality of OO Code", vol. 14(3), pp. 385-400, 2002.

[8]. Shyam R. Chidamber , Chris F. Kemerer , "A Metric suite for Object Oriented design", M.I.T. Solan School of Management E53-315, 1993.

[9]. Giancarlo Succi, Luigi Benedicenti and Martin Mintchev, "Defining Metrics for Software Components", Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. 10, pp. 16-23, 2000.

[10]. Washizaki, Yamamoto, Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components", Proceedings of the ninth international Conference on Software Engineering, pp. 211-213, 2003 .

[11]. Briand, L. C., Daly, J. W., and Wust, J. K., "A Unified Framework for Coupling Measurement in Object-Oriented Systems," IEEE Transactions on Software Engineering, vol. 25, pp. 91-121, January/February 1999.

[12]. Parul Gandhi, Pradeep Kumar Bhatia, "Reusability Metrics for Object Oriented System", An Alternative Approach International Journal of Software Engineering (IJSE), vol. 1, Issue 4, 2002.

[13]. Shyam R. Chidamber and Chris F. Kemerer, "A Metric suite of Object Oriented Design", IEEE Transaction on Software Engineering, vol. 20, No. 6, pp. 476-493, 1994.

[14]. Chidamber and Kemerer, "A suite of Metrics", Japan Advanced Institute of science and Technology School of Information Science, May 2008.

[15]. E. Chandra and P. Edith Linda, "Class Break Point Determination Using CK Metrics Thresholds", Global journal of Computer Science and Technology, Vol. 10 issue 14, November 2010, pp.-73-77.

[16]. Sonu Dhariwal and Pradeep Kumar Bhatia, "Design of COTS Metrics based upon CK Metrics", International Journal of Engineering Research & Technology (IJERT), Vol. 2, Issues 9, Sept. 2013, pp.-2096-2103.

[17]. Gaurav Kumar, Pradeep Kumar Bhatia, "Neuro-Fuzzy Model to Estimate & Optimize Quality and Performance of Component Based Software Engineering", ACM SIGSOFT Software Engineering Notes, Vol. 40, Issue 2, pp. 1-6, March 2015.

[18]. Gaurav Kumar, Pradeep Kumar Bhatia, "Optimization of Component Based Software Engineering Model Using Neural Network", BIJIT - BVICAM's International Journal of Information Technology, Vol. 6, No. 2, pp. 732-742, July - Dec. 2014.

[19]. Gaurav Kumar, Pradeep Kumar Bhatia, "Empirical Assessment and Optimization of Software Cost Estimation using Soft Computing Techniques", Advanced Computing and Communication Technologies, Publisher - Springer Singapore, Vol. 452, pp. 117-130, June 2016.